Keynotes and Question Answers for Semester-End Exam

Module I: Introduction (10%)

Key Topics:

1. Difference between C and C++:

C is a procedural programming language that follows a top-down approach, whereas C++ is an object-oriented programming language that follows the bottom-up approach and supports features like classes, inheritance, polymorphism, etc. C does not support data abstraction, encapsulation, and inheritance; C++ supports all these features.

2. Procedure-Oriented vs Object-Oriented Approach:

Procedure-Oriented: Focuses on functions and operations. Example: C. Object-Oriented: Focuses on objects that have both data and methods. Example: C++.

3. Basic Concepts:

Objects: Instances of a class that hold data and can perform operations.
Classes: Templates or blueprints for creating objects.
Abstraction: Hiding unnecessary details to focus on relevant information.
Encapsulation: Bundling the data and methods that operate on the data into a single unit (class).
Inheritance: Mechanism where a new class acquires properties of an existing class.
Polymorphism: Ability of an object to take many forms. Can be achieved through method overloading or overriding.
Dynamic Binding: Method calls are resolved at runtime.
Message Passing: Objects communicate by sending messages (calling methods).

4. Characteristics of Object-Oriented Languages:

Encapsulation, inheritance, polymorphism, abstraction, dynamic binding, and message passing are the key characteristics of OOP.

Module II: Classes & Objects (25%)

Key Topics:

1. Abstract Data Types: Data types that are defined by the programmer, encapsulating data and operations on that data.

2. Object and Classes:

Object: An instance of a class. Class: A blueprint that defines attributes (data) and methods (functions).

3. C++ Class Declaration:

```
class ClassName {

 public:

// data members

 // member functions

};
```

4. Local and Global Classes:

Local Class: Defined within a function.
Global Class: Defined outside of any function.

5. State, Identity, and Behaviour of an Object:

State: The values of an object's attributes.
Identity: A unique identifier for the object.
Behaviour: The functions (methods) an object can perform.

6. Scope Resolution Operator (::): Used to define methods outside the class definition.

7. Friend Functions: Functions that can access the private and protected members of a class.

8. Inline Functions: Functions defined with the inline keyword to reduce function call overhead.

9. Constructors and Destructors:

Constructor: Initializes an object when it is created.
Destructor: Cleans up when an object goes out of scope.

10. Static Class Data: Class data shared by all objects of the class.

11. Array of Objects: Arrays that can hold multiple objects of a class.

12. Constant Member Functions: Functions that do not modify the state of the object.

13. Memory Management Operators: new and delete operators for dynamic memory allocation and deallocation.

Module III: Inheritance (20%)

Key Topics:

1. Inheritance: Mechanism where a class derives properties and behaviors from another class.

2. Types of Inheritance:

Single Inheritance: One base class and one derived class.
Multiple Inheritance: A derived class inherits from multiple base classes.
Multilevel Inheritance: A class inherits from a derived class.
Hierarchical Inheritance: Multiple derived classes inherit from a single base class.

3. Access Modes:

Public: Members are accessible from outside the class.
Private: Members are not accessible from outside.
Protected: Members are accessible in derived classes.

4. Abstract Classes: Classes that cannot be instantiated and contain at least one pure virtual function.

5. Ambiguity Resolution: Resolved using scope resolution operator or virtual base class.

6. Aggregation and Composition:

Aggregation: 'Has-a' relationship, weak association.
Composition: Strong association, where the lifetime of the contained object depends on the lifetime of the container object.

7. Overriding Inherited Methods: Redefining base class methods in the derived class.

8. Constructors in Derived Classes: Constructors of derived classes call base class constructors.

9. Nesting of Classes: Defining a class inside another class.

Module IV: Polymorphism (20%)

Key Topics:

1. Polymorphism: Ability of a function or an object to take many forms.

Compile-time Polymorphism: Resolved during compilation (e.g., function overloading, operator overloading).
Runtime Polymorphism: Resolved during runtime (e.g., virtual functions).

2. Function Overloading: Multiple functions with the same name but different parameters.

3. Operator Overloading: Redefining the way operators work for user-defined types.

4. Polymorphism by Parameter: Same function name with different parameters.

5. Pointer to Objects: Pointers that can point to objects of a class.

6. This Pointer: Pointer that points to the current object.

7. Virtual Functions: Functions in base class that can be overridden in derived class to support dynamic dispatch.

8. Pure Virtual Functions: Virtual functions that do not have a definition in the base class and must be implemented by derived classes.

Module V: Strings, Files, and Exception Handling (25%)

Key Topics:

1. Manipulating Strings: Functions such as strlen(), strcpy(), strcat(), strcmp() for handling C-style strings, and std::string for C++ string handling.

2. Streams and File Handling:

ifstream: Used for reading files.
ofstream: Used for writing files.
fstream: Used for both reading and writing files.

3. Formatted and Unformatted I/O:

Formatted: cin, cout with format specifiers.
Unformatted: get(), put() for raw data input and output.

4. Exception Handling:

try, catch, throw: Used to handle exceptions and errors.
Standard Exceptions: std::exception, std::runtime_error, etc.

5. Generic Programming:

Function Template: Template to create generic functions.
Class Template: Template to create generic classes.

6. Standard Template Library (STL):

Containers: vector, list, set, map, etc.
Algorithms: Sorting, searching, etc.
Iterators: Used to traverse containers.
Other STL Elements: Allocators, function objects, etc.